A P System for Solving All-Solutions of TSP

Ping Guo, Junqi Xiang, Jingya Xie, Jinhang Zheng
College of Computer Science
Chongqing University
Chongqing, China

***Abstract*—P system is a parallel computing system based on a membrane computing model. Since the calculation process of the P system has the characteristics of maximum parallelism and Non-determinism, it has been used to solve the NP-hard problem in polynomial time. This paper designs a P system for TSP problem solving. This P system can not only determine whether the TSP problem has solution, but also give the all-solution when the TSP problem is solved. Finally, an example is given to illustrate the feasibility and effectiveness of the P system designed in this paper.**

***Keywords*—P system, TSP, membrane computing, natural computing**

I. Introduction

Membrane computing is a kind of biological calculation model which is inspired by living cell functions and tissues. Its information processing process adopts the parallelism and non-determinism of biochemical reaction in biological cells. The information processing system based on the membrane computing model is called the P system, which has the characteristics of computational parallelism and non-determinism. P systems have been studied can be divided into three categories: cell-like P system [1], [2], tissue-like P system [3], [4] and like-neural P system [5], [6].

Researchers have designed several P systems to solve NP-hard problems, such as SAT [7], [8], HPP [9], [10], TSP [11]-[15] and so on. The TSP (namely, travelling salesman problem) is a typical representative of the NP hard problem. To solve the TSP, researchers have proposed many algorithms for decades. According to whether the algorithm is to find the global optimal solution, these algorithms can be divided into two categories: exact algorithms and approximate algorithms. In [11], authors solves the symmetric TSP by using an improved branch and bound algorithm with a new lower bounds. In [12], authors proposes a novel ant colony optimisation (ACO) algorithm Moderate Ant System to solve TSP, this algorithm is experimentally turned out to be effective and competitive. In P system, some kinds of P system also have been proposed to solve the TSP. In [13], authors proposes a heuristic scheme with distributed asynchronous parallel computation for solving TSP problems, and the genetic algorithms are used to select the appropriate Hamiltonian path in each membrane. However, membranes used in this paper only are structures that hold programs and data, which don't conform to Gheorghe Păun's model of membrane computing. In [14], authors proposes a new type of approximate algorithms called membrane algorithms for solving TSP, a membrane algorithm borrows nested membrane structures and a number of sub-algorithms which can be any approximate algorithm for optimization problems

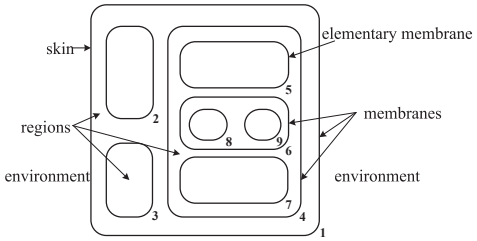

Fig. 1. The structure of cell-like P system.

are stored in membrane separated regions. Obviously, membrane algorithms don't conform to Gheorghe Păun's model of membrane computing too.

As a continuation of the research in [15], we have designed a P system to solve the TSP problem in this paper. The P system includes path construction, path detection, path comparison and path clipping, and its computational complexity is $O(n^2)$. The rest part of this paper is as follows: section II briefly introduces cell-like P system. In Section III, we design a parallel computing method which is suitable for P system to solve TSP problem. Section IV proposes a P system to solve TSP. The P system's structure and evolution rules are given, and the computational complexity of the P system is analyzed. In Section V, we give an example to show the process of solving the TSP using the P system designed in this paper. The conclusion is drawn in the last section.

II. Foundations

This paper is based on the cell-like P system. The cell-like P system is a class of the most basic P system, which consists of a series of membrane nesting, its structure shown in Fig. 1 [8]. A P system consists of a membrane structure, objects and evolutionary rules. The membrane structure consists of a skin, multiple membranes and multiple elementary membranes (in the absence of confusion, said the membrane). The region outside the skin is called the environment, which provides computing objects for the P system. The calculation objects (typically represented by the multiset of objects) and the object evolution rules are stored in the inner region of each membrane. The evolutionary rules within the membrane follow the maximum parallelism and non-determinism to make the object multisets evolve. When there is no any object multiset in the P system can be evolved, we call the calculation of the P system is over, and the results (expressed as object multisets) of the calculation are stored in a specific membrane or environment. If the evolution of the P system never stops, we call the calculation failed and no calculation results.

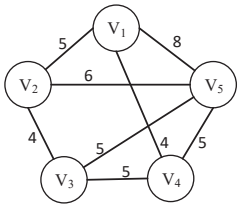

Fig. 2. Graph G.

According to [16], [17], the cell-like P system can be formally described as (1).

$$\Pi = (O, \mu, \omega_1, \cdots, \omega_m, R_1, \cdots, R_m, i_o) \tag{1}$$

Where,

1) O is the non-empty alphabet.$\forall o \in O$ is an object in Π. O^* is the Kleene closure over O, $\forall \omega \in O^*$ called an multiset in Π. Let λ is empty multiset, $O^+ = O^* - \{\lambda\}$;

2) μ is the membrane structure of Π. μ has m membrane, and each membrane is marked with a unique label i $(1 \leq i \leq m)$.

3) ω_i $(1 \leq i \leq m)$ is a multiset of objects placed in membrane i.

4) R_i $(1 \leq i \leq m)$ is the finite set of the evolution rules in membrane i of Π.

5) i_o is the label of a membrane to store the calculation results. Especially, $i_o = 0$ indicates that the output is stored in the environment of Π.

In Π, the maximal parallelism and Non-determinism of the rule execution mean:

1) Maximal parallelism: At any time, all rules can be executed must be performed at the same time.

2) Non-determinism: Suppose n rules are competing for execution, but P system can only support m $(m < n)$ rule execution, then m rules are randomly selected from n rules to execute.

III. TSP and the Parallel Algorithm

TSP is a NP-hard problem in combinatorial optimization, which can be described as: Given an undirected weighted graph $G = (V, E)$, where V is the vertex set and E is the edge set. For a given vertex v, find a path P that passes through all the other vertices once and only once and finally returns to the vertex v, and the sum of the weights on P (called the cost of P) is the smallest. In other words, the TSP is to find the Hamiltonian cycles with the least cost in all Hamiltonian circles of G.

Fig. 2 shows an example of an undirected weighted graph G. With V_1 as the starting and ending vertex, then the Hamiltonian cycles of G includes $\{\mathrm{V}_1 \to \mathrm{V}_2 \to \mathrm{V}_3 \to \mathrm{V}_4 \to \mathrm{V}_5 \to \mathrm{V}_1\}$, $\{\mathrm{V}_1 \to \mathrm{V}_2 \to \mathrm{V}_3 \to \mathrm{V}_5 \to \mathrm{V}_4 \to \mathrm{V}_1\}$, $\{\mathrm{V}_1 \to \mathrm{V}_2 \to \mathrm{V}_5 \to \mathrm{V}_3 \to \mathrm{V}_4 \to \mathrm{V}_1\}$ and so on. As we can see from Fig. 2, the minimum cost cycle is the second Hamiltonian cycle, so the solution of travelling salesman problem for Fig. 2 is $\{\mathrm{V}_1 \to \mathrm{V}_2 \to \mathrm{V}_3 \to \mathrm{V}_5 \to \mathrm{V}_4 \to \mathrm{V}_1\}$.

TABLE I. Algorithm: PATSP

Input: undirected weighted graph G=(V, E)and starting vertex v_0;
Output: the minimum cost cycle path or No;
(1) Path construction: Construct all legal paths in parallel, all paths make up a multi-tree, the steps of constructing one legal path P as follows: 1) Add v_0 to the path P as the common root node; 2) If there is edge $e =< v_i, v_j >$, v_i is the last vertex of path P and $v_j \notin P$, then add e and v_j to path P so that v_j becomes the last vertex of P; 3) Repeat step 2) until no vertex could be added to path P; 4) If all the vertices in graph G have been added to path P and there is an edge connecting the last vertex of path P to v_0,then add v_0 to path P as the last vertex; (2) Path detection: Delete illegal Hamiltonian cycle paths while constructing the paths: 1) If there is any vertex that cannot be added to path P, delete path P; 2) If the last vertex of path P is not v_0, delete path P; (3) Path comparison: Find a Hamiltonian cycle with minimum cost among all Hamiltonian cycles of G: 1) Starting from every leaf node to find the cost of every Hamiltonian cycle path; 2) If several paths share the common parent node, compare the cost of each path, find the path with minimum cost among them; 3) repeat 2) until the root node has been visited; (4) Path cutting: Delete paths that don't have the minimum weight; (5) Output: Output travelling salesman path or No.
End

In [15], a parallel algorithm PAHCP (Parallel algorithm for Hamiltonian cycle problem) is given to solve the all solution of the Hamiltonian problem. Based on the idea of PAHCP, a parallel algorithm PATSP (Parallel algorithm for TSP) for all solutions of TSP can be described as Table 1.

IV. Design of P Systems Π_{TSP}

In this section, we have designed a P system Π_{TSP} for solving TSP based on the algorithm which discussed in Section III.

A. The Definition of Π_{TSP}

As the cell-like P system just normally defined by (1), we defined this cell-like P system Π_{TSP} as follows:

$$\Pi_{\mathrm{TSP}} = (O, \mu, \omega, R, \rho, i_o) \tag{2}$$

where,

1) O is a finite and non-empty alphabet of objects, which includes:

- Some normal objects:

which indicate vertices in the undirected weighted graph: $\{a_i, e_i, u_i, p_i, q_i, f_i \mid 1 \leq i \leq \mathrm{n}\}$

- Some special objects:

-y, w, z: y indicates that all vertices have been visited; w, z means that Hamiltonian path has been found.

-λ: represents an empty multiset.

-δ: is an operation that means dissolving the current membrane to release the object to the outside of the membrane.

In addition, other objects in the system will be described when they are used.

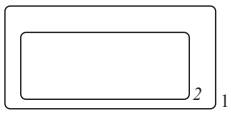

Fig. 3. The initial structure of the P system Π_{TSP}.

2) μ is the initial membrane structure of the system as shown in Fig. 3, which will change with the use of evolutionary rules.

3) ω=$\{\omega_1,\omega_2\}$ is the multiset in the initial membrane structure of Π_{TSP}. ω_1=$\{s^j \mid j = n\text{-}2\}\cup\{p_{io}, b, m, , f_{io}\}\cup\{a_i \mid 1\leq i\leq \mathrm{n}, i \neq i_o\}$, where n is the number of nodes in the graph, p_{io} means the output path will start from the node i_o ($1\leq i_o\leq$n), a_i represent the vertices of graph, m and ζ are used to control the execution of the rules. ω_2=λ.

4) R is the set of rules for system evolution, and $R = R^{\mathrm{C}} \cup R^{\mathrm{D}} \cup R^{\mathrm{F}} \cup R^{\mathrm{T}}$, where, R^{C} is used for path construction, R^{D} is used for path detection, R^{F} if used for path comparison and R^{T} is used for path cutting. Based on the Parallel algorithm PATSP, the procedure of applying the rules in Π_{TSP} is:

- path construction(see subsection B 1)).
- path detection (see subsection B 2)).
- path comparison (see subsection B 3)).
- path cutting (see subsection B 4)).

In Π_{TSP}, $\forall$ r $\in R_i$ has the following two forms:

•$(u \rightarrow v,\ k)$
•$(u \rightarrow v|_a,\ k)$

Where, $u \in O^+$,$v = v^{'}$ or $v = v^{'}\delta$, $v^{'} \in (O \times Tar)^*$, $Tar = \{here; out; in_j | 1 \leq j \leq m\}$ and $k \geq 1$.

(a) k indicates the priority, the smaller value k is set, the higher the priority of the corresponding rule is. High-priority rules will be executed before the lower-level rules.

(b) Tar identifies the location where the evolutionary results are stored. Here means v is remained in membrane i, out means v goes out of membrane i, and in_j means v goes to inner membrane j. To simplify the representation, here will be omitted.

(c) Object a is a promoter, it means the rule can only be applied in the presence of object a.

5) When the system halts, we will find the final result in membrane 1(i_0 corresponds to membrane 1 in Π_{TSP}).

B. The rules in Π_{TSP}

1) Path construction:

When Π_{TSP} starts, objects in skin membrane represent the undirected weighted graph: 1) a_i represents the vertices of graph G; 2) e represents the end of inputting vertices; 3) in a Hamiltonian cycle, f_i represents the starting vertex and the end one.

a) Visit vertex

To solve TSP, we firstly need to find all Hamiltonian cycle paths. That means we should visit from the starting vertex to all other vertices exactly once, then back to the starting vertex at last. In the beginning, the length of current path P is 0 because there is no vertex visited, then the length will increase by 1 if a vertex has been visited. The process are defined by rules in R^{C} ($1\leq i\leq$n, $1\leq j\leq$n, $1\leq k\leq$n):

r_1: $([ba_i]_k \rightarrow [e_i c[tp_i]_{k+1}]_k,\ 1)$
r_2: $(u_i \rightarrow u_i(a_i,\ in)|_c,\ 2)$
r_3: $(a_i \rightarrow a_i(a_i,\ in)|_c,\ 2)$
r_4: $(s \rightarrow s(s,\ in)|_c,\ 2)$
r_5: $(f_i \rightarrow f_i(f_i,\ in)|_c,\ 2)$
r_6: $(m \rightarrow m(m,\ in)|_c,\ 2)$
r_7: $(r \rightarrow r(r,\ in)|_c,\ 2)$
r_8: $(\zeta \rightarrow \zeta(\zeta,\ in)|_c,\ 2)$
r_9: $(cp_i \rightarrow p_i b(q_i\tau, in),\ 3)$
r_{10}: $(e_i \rightarrow u_i\ |_c,\ 2)$
r_{11}: $(ts \rightarrow \lambda,\ 2)$
r_{12}: $(q_i p_j \rightarrow p_j b r^n,\ 1)$
r_{13}: $(q_i p_j \rightarrow d,\ 1)$

We use rule r_1 to create sub-membrane which can determine whether there is an edge between two vertices, r_2∼r_8 are used to copy objects and transfer them to new membrane, r_2∼r_8 are executed for the determination whether add the new vertex to current path P. if there exist an edge between the last vertex of current path P to the new vertex, r_{12} is executed,and n is the weight of the edge, otherwise, r_{13} is applied, which means there is no edge from those two vertex.

If V_i is the last vertex of current path P, and V_j is the vertex being visiting. Firstly, r_1∼r_8 is used to create sub-membrane, copy and transfer objects to the new sub-membrane. If there exists an edge from V_i to V_j, the rule r_{12} will be executed to create object b and r(the number of object r represents the weight of corresponding edge), and this means that V_j will be added to the current path. If there is no edge from V_i to V_j, r_{13} will be executed and object d will be created to dissolve the sub-membrane and objects in it, which means that is not a Hamiltonian cycle path.

b) Back to the starting vertex f_i

When all the vertices have been added to the path P, if V_j is the last vertex of path P, and there exist an edge from V_j to f_i, the path P is a Hamiltonian cycle path. And if there is no edge from V_j to f_i, the path P is not a Hamiltonian cycle path. The process are defined by rules in R^{C} ($1\leq i\leq$n):

r_{14}: $(bf_i \rightarrow y[p_i],\ 3)$
r_{15}: $(yb \rightarrow (o; w,\ out),\ 1)$
r_{16}: $(yd \rightarrow d\delta,\ 1)$
r_{17}: $(p_i y \rightarrow p_i(yq_i,\ in),\ 1)$
r_{18}: $(rp_i \rightarrow p_i(r_i,\ out)|_o,\ 1)$

After the execution of r_{14}, a new sub-membrane and object p_i (represent the starting vertex) will be created, object q_i (represent the last vertex of path P) will be created and sent into the sub-membrane with the execution of r_{17}. At this time, r_{12} will be executed to create object b and object r if there is an edge from the last vertex to the starting one. Then r_{15} will be executed to create object object o and send object$w$ (indicate that there is a Hamiltonian cycle path) to outer membrane,and because of the existence of object o, all object r will be converted to r_i to outer membrane. r_{13} will be executed to create object d if there is no edge from the last vertex to the starting vertex. The object d will cause the execution of the r_{16}, which dissolve the sub-membrane and shows that path P cannot be a Hamiltonian cycle path.

2) Path detection:

By detecting, it is judged whether the newly generated membrane is a valid membrane on the Hamiltonian path, and

if it is not then pruning it.

a) Judgment

When rule r_{15} in R^{C} is applied, object w will be created to send to outer membrane, and it shows that a Hamiltonian cycle path has been found. Rules in R^{D} associated with the process are:

r_{19}: $(w \to z\delta, 1)$ r_{23}: $(vzh \to v(t,out), 4)$
r_{20}: $(tz \to v(t, out), 3)$ r_{24}: $(szh \to z, 2)$
r_{21}: $(szt \to vz, 1)$ r_{25}: $(zh \to k, 5)$
r_{22}: $(k \to h\delta, 1)$

Rule r_{19} is used to reduce the thickness of membrane and it can covert object w to object z. The existence of object s means that there are some sub-membrane not disposed in current membrane. r_{20} is used to create object v and send object t to outside when there is no object s. the number of vertices in current path is represented by the depth of membrane. object p_i represents vertex V_i is on the current path, and the number of v represents the number of Hamiltonian cycle paths. If there is object s in current membrane, r_{21} will be executed to create object v only.

If there is no Hamiltonian cycle path found, rule r_{22} will be executed to send object h(shows that no Hamiltonian cycle path was found) to outside. r_{24} will be executed if object s exists in outer membrane; if there is only object v exist in outer membrane, it shows that all sub-membrane have been disposed and there is Hamiltonian cycle path exist, and r_{23} will be executed to delete object h and to send t to outside; if there is no object s and object v, it means that all sub-membrane have been disposed and no Hamiltonian cycle path was found, then r_{25} will be executed to create object k for the next step.

b) Pruning

After path detection, we need to remain the meaningful membranes and objects which shows the found Hamiltonian cycle path and to abandon the useless membranes and objects In the following cases, pruning is required in Π_{TSP}:

i) Visiting each vertex. Let V_i be the next vertex to be visited, and we need to find out whether there is an edge from the last vertex in current path to V_i, we just create a new sub-membrane for this process by rule r_1 in R^{C}. If there is no edge the sub-membrane and the objects in it will be dissolved by those delete rules in R^{D}. The rules in R^{D} associated with the process are:

r_{26}: $(s \to \lambda\,|_d, 1)$ r_{31}: $(d \to k\delta, 2)$
r_{27}: $(t \to \lambda\,|_d, 1)$ r_{32}: $(p_i \to \lambda\,|_k, 1)$
r_{28}: $(a_i \to \lambda\,|_d, 1)$ r_{33}: $(\zeta \to \lambda\,|_k, 1)$
r_{29}: $(u_i \to \lambda\,|_d, 1)$ r_{34}: $(m \to \lambda\,|_k, 1)$
r_{30}: $(f_i \to \lambda\,|_d, 1)$ r_{35}: $(r \to \lambda\,|_d, 1)$

ii) All sub-membrane have been created. If all sub-membrane have been created in current membrane, we need to delete objects in current membrane except s, p_i and reduce the thickness of current membrane. The rules in R^{D} associated with the process are ($1 \le i \le n$):

r_{36}: $(bu_i \to g, 2)$ r_{39}: $(f_i \to \lambda\,|_g, 1)$
r_{37}: $(a_i \to \lambda\,|_g, 1)$ r_{40}: $(g \to z\delta, 2)$
r_{38}: $(u_i \to \lambda\,|_g, 1)$

r_{36} is used to create new membrane with u_i representing vertex v_i has been added to current path. When there is no object a_i ($1 \le i \le n$) in current membrane, it means all vertices have been visited, and r_{36} with a lower priority will be executed to create object g. Then the delete rules will be executed to delete relative objects and membranes.

iii) All vertices have been added to current path. The next step is to determine if there is an edge from the last vertex to the starting one. if not, object d will be created by r_{13} in R^{C}. Then r_{31} and r_{22} in R^{D} will be executed to dissolve the sub-membrane and objects, and the current path is not a Hamiltonian cycle path. The rules in R^{D} associated with the process are ($1 \le i \le n$):

r_{31}: $(d \to k\delta, 2)$ r_{22}: $(k \to h\delta, 1)$
r_{32}: $(p_i \to \lambda\,|_k, 1)$

iv) No Hamiltonian cycle path found after all sub-membranes were detected. In this case, we just dissolve the current membrane by the following rules in R^{D} ($1 \le i \le n$):

r_{32}: $(p_i \to \lambda\,|_k, 1)$ r_{22}: $(k \to h\delta, 1)$
r_{25}: $(zh \to k, 5)$

3) Path comparison:

When all Hamiltonian cycle paths have been constructed, we need to find a path with a minimum cost among all Hamiltonian cycle paths. Starting from the innermost membrane to skin membrane, we move object r_i($1 \le i \le n$) whose number represent the cost of one Hamiltonian cycle path to outer membranes and compare different paths to find a path with a minimum cost. Rules in R^{F} associated with this process ($1 \le i \le n$, $1 \le j \le n$):

r_{41}: $(r_i m \to c_i r_i \alpha_i, 1)$ r_{46}: $(\beta_i \to \lambda\,|_{yi}, 1)$
r_{42}: $(r_i \to \beta_i|_{ci}, 1)$ r_{47}: $(\alpha_i \to \lambda\,|_{yi}, 1)$
r_{43}: $(c_i v \to m|_{\beta i}, 1)$ r_{48}: $(\gamma \to \lambda|_{yi}, 1)$
r_{44}: $(\beta_i\beta_j \to \beta\gamma, 2)$ r_{49}: $(\beta\zeta\alpha_i \to \beta\zeta_i, 2)$
r_{45}: $(\beta_i\gamma \to y_i, 1)$ r_{50}: $(\beta \to \beta_i\,|_{\zeta i}, 1)$

The strategy of our comparison is pairwise comparison, rule of type r_{41} and r_{42} is used to control that only two Hamiltonian cycle paths are compared every time. Because of the uniqueness of object m in a membrane, object r_i will be converted to object β_i sequentially. The number of object r_i and r_j represents the cost of two different Hamiltonian cycle paths (path i and path j), the subscript of object r is decided by the subscript of object p in the corresponding sub-membrane. When object r_i and r_j has been converted to β_i and β_j by applying rule r_{42}, rule of type r_{44} will be used to convert β_i and β_j to β. Assume that the number of β_i is less than β_j, which means that the cost of path i is smaller than path j. So after rule r_{44} is applied, β_j will be left. Then rule of type r_{45}, r_{46} and r_{47} will be applied to delete α_j and all of object β_j. What's more, rule of type r_{48} will be applied to delete object γ.

After all object r_j has been deleted, we need to convert β to β_i for letting the process of comparison continue. When rule of type r_{49} is applied, α_i will be dissolved and ζ will be converted to ζ_i. And because the existence of object ζ_i, β will be converted to object β_i under the application of rule r_{50}. By now, a pairwise comparison has completed, object r_j which represent the larger cost of a Hamiltonian cycle path has been all deleted. Rules in R^{P} will applied until all object r_k ($1 \leq k \leq n$) which don't represent the Hamiltonian cycle path a minimum cost in the membrane are deleted.

4) Path cutting:

When a Hamiltonian cycle path has been detected that doesn't have a minimum cost, we need to delete corresponding membranes that represent this Hamiltonian cycle path. Rules in R^{T} associated with the process are ($1 \leq i \leq n$, $1 \leq j \leq n$):

r_{51}: $(y_i \rightarrow (y_i, \text{in})|_{\neg \alpha i}, 1)$ r_{54}: $(xp_i \rightarrow d(x, \text{in}), 1)$
r_{52}: $(y_i p_i \rightarrow p_i x, 1)$ r_{55}: $(\beta_i \rightarrow n_i \,|_{\neg v}, 1)$
r_{53}: $(y_i p_j \rightarrow (y_i, \text{out}), 1)$ r_{56}: $(n_i \rightarrow (r_j, \text{out})|_{pj}, 1)$

Object y is used to start delete rules, the subscript of object y is decided by the subscript of object p in the corresponding sub-membrane. When object y_i exists and α_i is dissolved, a Hamiltonian cycle path has been detected that doesn't have a minimum cost. Then under the application of rule r_{51}, r_{52} and r_{53}, object y_i will get in and out sub-membranes continuously until the subscript of object p in a membrane is the same as y_i. After rule r_{52} in sub-membrane is applied, object x will be created. When object x exists, rule r_{54} will be applied to create object d and send object x into sub-membrane. Because of the existence of d, the membrane and objects in it will be dissolved. With the implementation of rule r_{54}, all corresponding sub-membranes will be dissolved. Once object v doesn't exist in membrane, path comparison in this membrane has completed, we need to move object r_i to outer membrane. Rule r_{55} will convert β_i to n_i when object v doesn't exist in membrane, then all object n_i will be convert into r_j and be sent out because of the existence of object p_j.

C. Parallelism Analysis of Π_{TSP}

1) Analysis of Π_{TSP}: For a complete undirected weighted graph with n vertices, we can see that the number of all possible Hamiltonian cycle path is at most n! by using the exhaustive method. So as long as taking n! case into account and judging that whether each case can constitute a ring, we can find all the Hamilton loop. As is shown in Fig. 4, this process could be described as construct a multi-tree where each possible Hamiltonian cycle path could be found. When the P system starts, let the starting vertex be V_1, in the outermost membrane is the objects represent the rest vertices $V_2 \sim V_n$. For every V_i ($2 \leq i \leq n$) hasn't been visited, we create a new membrane to judge whether there is an edge from V_1 to V_i (corresponding rule is r_1). If the two vertices are connected, the new sub-membranes will be remained (corresponding rules are $r_2 \sim r_8$). Then for the n-2 vertices that haven't been visited, n-2 new membranes will be created continuously in just sub-membranes. The process will be repeated until each vertex is on the path. What can be summarized by the above process is n! case could all be taken into consideration. For each of the generated path, we judge whether there is an edge between the two vertices in the new sub-membrane (corresponding rules are r_{12} and r_{13}). If there is an edge connected between two vertices, then the sub- membrane will be remained; otherwise, we need to dissolve surplus membranes and pruning is needed in four situations in Π_{TSP} (crucial corresponding rules are r_{13}, r_{22}, r_{31} and r_{40}). So when the process of path detection is completed, only membranes that represents Hamiltonian cycle paths will be remained.

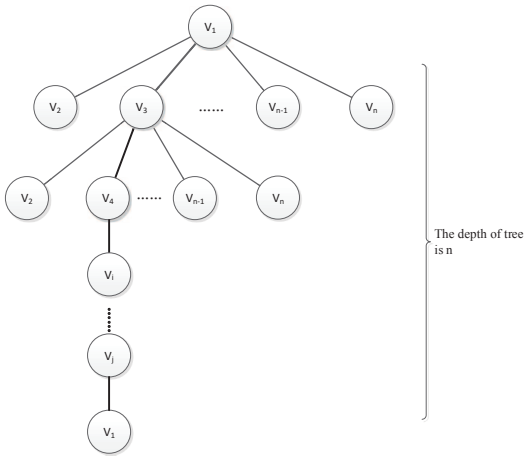

Fig. 4. The process of constructing a multi-tree.

Hamiltonian cycle paths are represented by a series of membranes that are nested one by one in our P system. As described in algorithm PATSP, all Hamiltonian cycle paths constitutes a multi-tree together. Because each leaf node represents a Hamiltonian cycle paths, so to find the solution of travelling salesman problem, we only need start from the leave nodes of the multi-tree to compare the weight of each Hamiltonian cycle path until we find the Hamiltonian cycle path with minimum weight. In our P system, starting from the innermost membranes, then compare the weight of each Hamiltonian cycle path (corresponding rules are $r_{41} \sim r_{44}$) and delete the the corresponding membrane structures (corresponding rules are $r_{45} \sim r_{48}$ and $r_{51} \sim r_{54}$) represents Hamiltonian cycle paths without a minimum weight. What's more, transfer the weight of the path has a bigger weight to outer membrane (corresponding rules are $r_{55} \sim r_{56}$) and continue the process of comparison until we find the path with a global minimum weight. What can be summarized by the above process is the right result will be generated when the whole system halts.

2) Analysis of time complexity: According to the maximum parallelism of P systems, the rules that meet their requirements will be executed at the same time. As shown in Fig. 5, it is the process of the execution of rules in Π_{TSP}. We assume that the time cost for executing a rule is a slice. What's more, rules like $[r_i]$ means r_i could be executed or not in Fig. 5.

a) Cost of path construction

The process of path construction is to use the parallel computing methods to construct every possible Hamiltonian cycle path. For a complete undirected weighted graph with n vertices, the number of vertex on a Hamiltonian cycle path is n too. For the i^{th} vertex on the path, there are n-i vertices that should be taken into consideration ($n - i + 4$ slices). So the whole process will take $\sum_{i=1}^{n}(n - i + 4)$ slices.

b) Cost of path detection

The process of path detection and path construction happens at the same time. Dissolving membranes that represent one illegal Hamiltonian cycle path cost constant time (up to 3 slices). Because path construction and path detection happens parallel, so it cost about 3×n slices in total.

c) Cost of path comparison

One comparison costs 8 slices. Because the process of path comparison is parallel and starts from the innermost membranes. For an undirected weighted graph with n vertices, the depth of membranes is n. So it cost 8n slices in total.

d) Cost of path cutting

The process of path cutting happens at the same time with path comparison. One path cutting costs 3 slices. When the depth of membranes is n, the total cost of path cutting is 3n slices. In summary, the total cost of Π_{TSP} can be computed as follows: T_{TSP}=$\sum_{i=1}^{n}(n-i+4)+3\times n+8\times n+3\times n=\frac{1}{2}n(n-1)+18n=O(n^2)$.

In [17], author uses RanGen (Randomly Generating) MCGA (Membrane-Computing-Genetic-Algorithm) to solve travelling salesman problem, the time complexity of the algorithm is O(n^3) time. This computation is much faster than that of brute force complete enumeration method in serial, but is still slower than PATSP algorithm. Compared with the traditional ant colony algorithm and genetic algorithm, our algorithm is not only better in time complexity, but also can solve the exact solution of the problem.

V. Calculate Instance

In this section, An example is given to show the whole process to solve TSP in Π_{TSP}.

The undirected weighted graph G=(V, E) is shown in Fig. 2, let V_1 be the starting vertex (also the last vertex). The process in Π_{TSP} are as followed:

A. Path Construction

Objects represent the undirected weighted graph which should be input to the skin membrane. Firstly, input multiset $p_1a_2a_3a_4a_5$, then input f_1, last input $mbs_3\zeta$. We will construct legal paths by membrane creation. The available rules in R^{C} are applied in the order of $\{\mathrm{r}_1\}\rightarrow\{\mathrm{r}_2\sim\mathrm{r}_8\}\rightarrow\{\mathrm{r}_9\sim\mathrm{r}_{11}\}\rightarrow\{\mathrm{r}_{12}\}$. There is multiset $s^3p_1ba_2a_3a_4a_5f_1m\zeta$ in membrane 2, rule r_1 is used to create sub-membrane and $\mathrm{r}_2\sim\mathrm{r}_8$ are used to copy objects and transfer them to new sub-membrane. At first, the length of current path is 1, and object p_1 shows that the vertex V_1 has been added to current path. Then $\mathrm{r}_1\sim\mathrm{r}_8$ will be executed to create a new sub-membrane with multiset $s^3tq_1p_2a_3a_4a_5f_1m\tau\zeta$ to determine if there are edges from V_1 to V_2, V_3, V_4, V_5. As shown in Fig. 2, there exist an edge from V_1 to V_2. So rule r_{12} in R^{C} will be executed, multiset q_1p_2 are converted to p_2br^5 and the new sub-membrane will not be dissolved, which means that V_2 has been added to current path. There are same process when disposing V_4 and V_5, because there are edges from V_1 to them. And the sub-membrane will be dissolved when disposing V_3 because there is no edge from V_3 to V_1. Objects in sub-membrane continue to evolve, and there is multiset $s^2p_2ba_3a_4a_5f_1\tau$ in the new sub-membrane

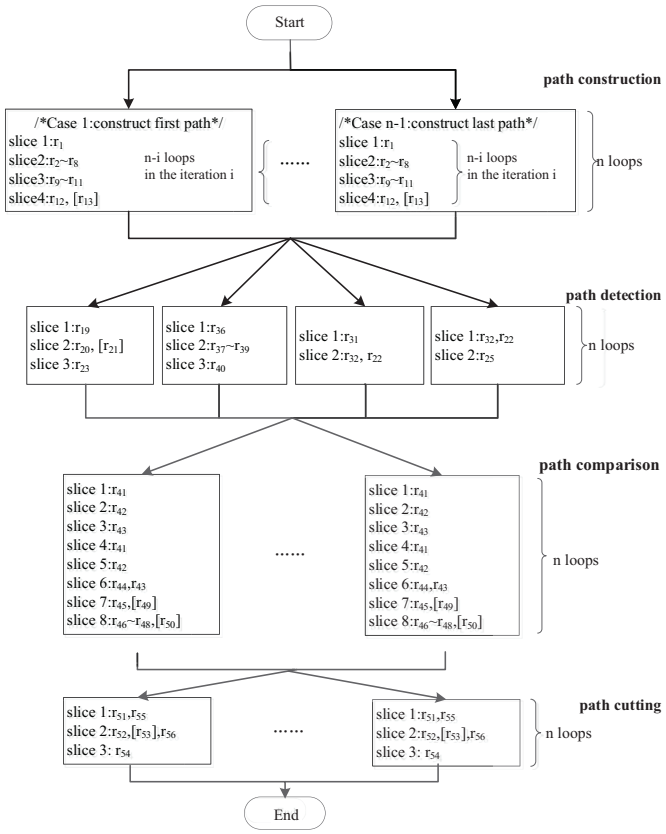

Fig. 5. the process of execution of the rules in Π_{TSP}.

when disposing V_2, which shows that V_2 is the last vertex in current path and V_3, V_4 and V_5 have not been visited. So the next step is continue to create new sub-membranes to visit V_3, V_4 and V_5.

If all vertices have been visited and added to current path P, the next step is to determine whether there is an edge from the last added vertex to the first one of path P. After we have added V_3,V_4 and V_5 to current path P (because there are edges connecting them), we consider the edge from V_5 to V_1, so the rule r_{14} and r_{17} in R^{C} is applied. V_1 will be added to current path P with the execution of r_{12} and path P is a Hamiltonian cycle path.

B. Path Detection

1) Judgment: As shown in Fig. 6, after r_{15} in R^{C} are applied, object w will be created and sent to membrane 5 which means there is a Hamiltonian cycle path found. And now there is a multiset $wtp_5m\tau\zeta r^{27}$ in membrane 5, after r_{19} in R^{D} is executed the multiset in membrane 5 will change to $mtp_5z\zeta r^{27}$. Then r_{20} in R^{D} will be applied to create object v and send object t to membrane 4. The rule r_{21} will be executed to evolve multiset szt to vz.

As shown in Fig. 7, when object d is created in membrane 6, which means there is no Hamiltonian cycle path. After that, membrane 6 will be dissolved by r_{16} and object d will be sent to membrane 5. Then object d will be converted to object k and the thickness of membrane 5 will be reduced by the execution

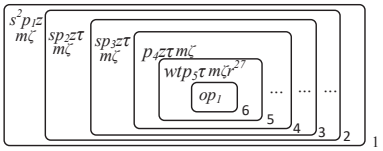

Fig. 6. Exist a Hamiltonian cycle path.

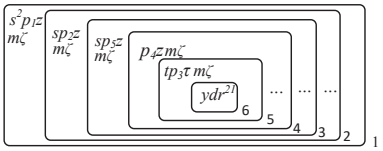

Fig. 7. No Hamiltonian cycle path.

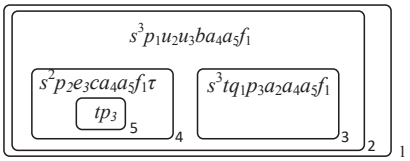

Fig. 8. Visit vertex of graph G.

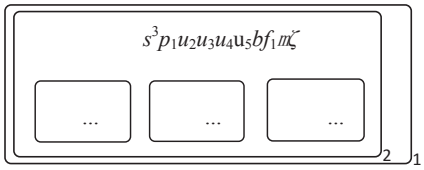

Fig. 9. All object a_i has evolved to u_i in membrane 2.

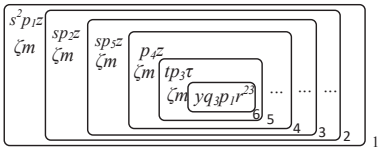

Fig. 10. Each vertex has been added to current path.

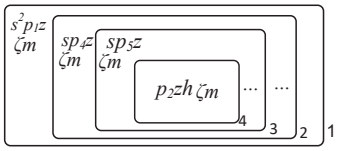

Fig. 11. All sub membranes of membrane 4 have been disposed.

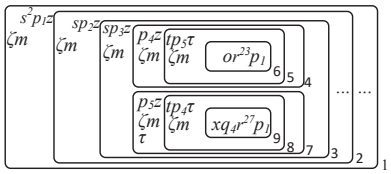

Fig. 12. Two Hamiltonian cycle paths.

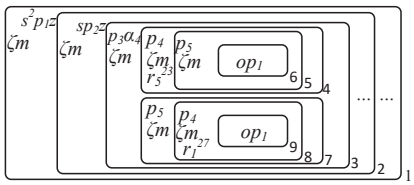

Fig. 13. The membrane structure in path comparison.

of r_{31} in R^{D}. Because of the existence of object k, rule r_{32} in R^{D} will be applied to dissolve object p_3 in membrane 5.

2) Pruning: The process of pruning is to dissolve the surplus membranes and objects and remain the meaningful membranes and objects which indicate the Hamiltonian cycle path. The associated execution of rules in this example are as followed:

a) As shown in Fig. 8, because there is no edge from V_1 to V_3, rule r_{13} is applied to create object d. Then membrane 3 and all objects in it will be dissolved by the execution of rules in R^{D}.

b) As shown in Fig. 9, all objects a_i has evolved to u_i which means that all vertices have been visited. Then rules in R^{D} are applied in the order of $\{\mathrm{r}_{36}\}\rightarrow\{\mathrm{r}_{37}, \mathrm{r}_{38}, \mathrm{r}_{39}\}\rightarrow\{\mathrm{r}_{40}\}$. With the execution of those rules, objects in membrane 2 will be dissolved except s, p_i.

c) As shown in Fig. 10, all vertices have been added to current path P. However, there is no edge from V_3 to V_1, so r_{13} in R^{C} is applied to dissolve membrane 6 and object d will be sent into membrane 5, then object d will evolve to object k due to the execution of rule r_{31} in R^{D}.

d) As shown in Fig. 11, there is no Hamiltonian cycle path found and all sub-membranes of membrane 4 have been disposed. With the execution of rule r_{25} and r_{32} in R^{D}, membrane 4 and objects p_2 in it will be dissolved.

C. Path Comparison

As shown in Fig. 12, two Hamiltonian cycle paths has been found in membrane with the cost of 23 and 27. We need to find the smaller one between the cost of two Hamiltonian cycle paths. Because of using of rule r_{18} in R^{C}, all object r will be sent out from the innermost membrane. After rule r_{19} and r_{20} in R^{D} is applied in membrane, the number of object v in membrane 5 is 1, after rule of type r_{41} and r_{42} in R^{F} is applied, all object r_1 has been converted to object β_1. Then rule of type r_{55} and r_{56} will be applied to convert object β_1 to object r_5 and send all object r_5 to membrane 4. Similar to the application of rules in membrane 5, all object r_5 will be converted to object r_4 and will be sent into membrane 3. By now, the membrane structure is shown in Fig. 13.

Rule of type r_{41} in R^{F} is used to create object c_4 which is used to convert all object r_4 to β_4 and because of the existence of object β_4, object c_4 will be converted to object m. As is shown in Fig. 14, all object r_4 in membrane 7 will also be sent into membrane 3 and will be converted to object r_5. After rule of type $\mathrm{r}_{41}\sim\mathrm{r}_{43}$ in R^{F} is used, object r_5 will be converted to object β_5, then the comparison of the cost of two Hamiltonian cycle path will start. After rule of type rule r_{44} in R^{F} is applied, object β_4 and object β_5 are converted to β. And three object are left in membrane 3. So rule of type r_{45} will be used next to create object y_5 which is used to delete object β_5, α_i, and γ. By now, objects r_5 which represents the larger cost of two

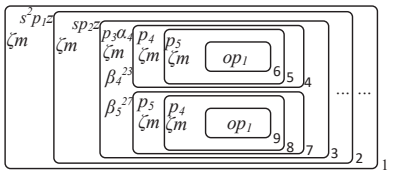

Fig. 14. The membrane structure in path comparison.

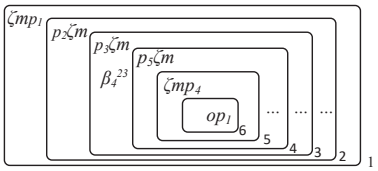

Fig. 15. The membrane structure after path comparison.

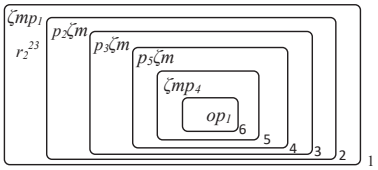

Fig. 16. The final membrane structure.

Hamiltonian cycle paths has been all deleted. Rule of type r_{49} in R^{F} is used to create object ζ_4 which is used to convert β to β_4. By now, a path comparison has been completed which is shown in Fig. 15.

D. Path Cutting

After a path comparison, we have known membranes and objects which represent a Hamiltonian cycle path with a larger cost. As shown in Fig. 15, membrane 7 and its sub-membranes should be dissolved. Object α_5 has been deleted because it represent the path with a larger cost. By applying the rule of type r_{51}∼r_{53} in R^{T}, object y_5 will be sent into a sub-membrane which has object p_5. Then by applying rule of type r_{54} in R^{T} continuously, object d will be created to start delete rules. As a result, corresponding membranes and objects will be dissolved. What's more, β_4 will be converted to n_4 by applying rule of type r_{55} in R^{T} because object v has been all dissolved which means that a path comparison has been completed. Then n_4 will be converted to object r_3 and will be sent to outer membrane to start a new path comparison because of the existence of object p_3. By now, a path cutting has been completed.

E. Final Result

When the whole system halts, the final membrane structure is shown in Fig. 16. As we can see in Fig. 16, only membranes and objects that represent the Hamiltonian path with a minimum cost are remained. Object p_i represents vertex v_i in graph. By detect object p_i in each membranes, we knows the path is: $\{\mathrm{V}_1 \rightarrow \mathrm{V}_2 \rightarrow \mathrm{V}_3 \rightarrow \mathrm{V}_5 \rightarrow \mathrm{V}_4 \rightarrow \mathrm{V}_1\}$.

VI. Conclusions

The cell-like P system is a new computational system inspired by biological cell behavior. This paper presents a cell-like P system Π_{TSP} to solve travelling salesman problem with $O(n^2)$ complexity. In Π_{TSP}, we firstly construct all Hamiltonian cycle paths by membrane creation, then find the Hamiltonian cycle path with a minimum cost, lastly remove all membranes and objects that do not contain the TSP solution. Finally, an example is given to illustrate the feasibility and effectiveness of our P system.

References

[1] A.Vitale, G.Mauri, C.Zandron. Simulation of a bounded symport/antiport P system with Brane calculi[J]. Biosystems, 2008, 91(3): 558-571.

[2] C.Martin-Vide, Gh. Păun, A. Rodríguez-Patón. On P systems with membrane creation[J]. Computer Science Journal of Moldova, 2001, 9(2): 134-145.

[3] C.Mart, Gh. Păun, J.Pazos. Tissue P systems[J]. Theoretical Computer Science, 2003, 296(2): 295-326.

[4] R. Freund, Gh. Păun, M. J. Pérez-Jiménez. Tissue P systems with channel states[J]. Theoretical Computer Science, 2005, 330(1): 101-116.

[5] X. Y. Zhang, X. X. Zeng, B. Luo, and J. B. Xu, Several Applications of Spiking Neural P Systems with Weights, Journal of Computational and Theoretical Nanoscience, 2012, 9(6): 769-777.

[6] T. Song, Y. Jiang, X. L. Shi, and X. X. Zeng, Small Universal Spiking Neural P Systems with Anti-Spikes, Journal of Computational and Theoretical Nanoscience, 2013, 10(4): 999-1006.

[7] P. Guo, J.-F. Ji, H.-Z. Chen, R. Liu, Solving All-SAT Problems by P Systems, Chinese Journal of Electronics, 2015, 24(4): 744-749.

[8] P. Guo, J. Zhu, M. Q. Zhou, A family of uniform P systems for All-SAT problem, Journal of Computational and Theoretical Nanoscience, 2016, 13(1): 319-326.

[9] L. Pan, A. Alhazov. Solving HPP and SAT by P systems with active membranes and separation rules[J]. Acta Informatica, 2006, 43(2):13 l-145.

[10] K Ishii, A. Fujiwara, Asynchronous P systems for SAT and Hamiltonian Cycle Problem, in: 2010 Second World Congress on Nature & Biologically Inspired Computing, IEEE, 2010: 513-519.

[11] M. Padberm, G. Rinaldi. A Branch-And-Cut Algorithm For The Resolution Of Large-Scale Symmetric Traveling Salesman Problems. Society for Industrial and Applied Mathematics, 1991, 33(1): 60-100.

[12] P. Guo, Z. J. Liu. Moderate ant system: An improved algorithm for solving TSP[C]. 7^{th} International Conference on Natural Computation, pp. 1190-1196, 2011.

[13] P. Manalastas. Membrane Computing with Genetic Algorithm for the Travelling Salesman Problem. In: Nishizaki S., Numao M., Caro J., Suarez M.T. (eds) Theory and Practice of Computation. Proceedings in Information and Communications Technology, vol 7: 116-123. Springer, Tokyo. 2013.

[14] T. Y. Nishida, Membrane Algorithms: Approximate Algorithms for NP-Complete Optimization Problems. Springer Berlin Heidelberg, 2006: 303-314.

[15] P. Guo, Y. L. Dai, H. Z. Chen, A P system for Hamiltonian cycle problem, Optik, 2016, 127(20): 8461-8468.

[16] Gh. Păun, Computing with membranes. Journal of Computer and System Sciences, 2000, 60(1): 108-143.

[17] Gh. Păun, Membrane Computing. An Introduction, Springer-Verlag, Berlln, 2002.